NASA/TM—2000-209956

ARL–TR–2262

NPSS on NASA's Information Power Grid: Using CORBA and Globus to Coordinate Multidisciplinary Aeroscience Applications

Isaac Lopez
U.S. Army Research Laboratory, Glenn Research Center, Cleveland, Ohio

Gregory J. Follen
Glenn Research Center, Cleveland, Ohio

Richard Gutierrez
Integral Systems, Cleveland, Ohio

Ian Foster, Brian Ginsburg, Olle Larsson, Stuart Martin, Steven Tuecke, and David Woodford
Argonne National Laboratory, Argonne, Illinois

Prepared for the
Computational Aerosciences Workshop
sponsored by the High Performance Computing and Communications Program
Moffett Field, California, February 15–17, 2000

National Aeronautics and
Space Administration

Glenn Research Center

July 2000

Acknowledgments

Research activities at Argonne National Laboratory relating to NPSS and Commodity Grid Toolkits were supported by NASA's High Performance Computing and Communications Program, by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Computational and Technology Research, U.S. Department of Energy, under Contract W-31-109-Eng-38, and by the National Science Foundation. The authors express appreciation to management of the High Performance Computing and Communications Program and to NASA Glenn Research Center's NPSS team.

Available from

NASA Center for Aerospace Information
7121 Standard Drive
Hanover, MD 21076
Price Code: A03

National Technical Information Service
5285 Port Royal Road
Springfield, VA 22100
Price Code: A03

Available electronically at http://gltrs.grc.nasa.gov/GLTRS

NPSS ON NASA'S INFORMATION POWER GRID: USING CORBA AND GLOBUS TO COORDINATE MULTIDISCIPLINARY AEROSCIENCE APPLICATIONS

Isaac Lopez
U.S. Army Research Laboratory
National Aeronautics and Space Administration
Glenn Research Center
Cleveland, Ohio 44135

Gregory J. Follen
National Aeronautics and Space Administration
Glenn Research Center
Cleveland, Ohio 44135

Richard Gutierrez
Integral Systems
Cleveland, Ohio 44135

Ian Foster, Brian Ginsburg, Olle Larsson, Stuart Martin, Steven Tuecke, and David Woodford
Argonne National Laboratory
Argonne, Illinois 60439

SUMMARY

This paper describes a project to evaluate the feasibility of combining Grid and Numerical Propulsion System Simulation (NPSS) technologies, with a view to leveraging the numerous advantages of commodity technologies in a high-performance Grid environment. A team from the NASA Glenn Research Center and Argonne National Laboratory has been studying three problems: a desktop-controlled parameter study using Excel (Microsoft Corporation); a multicomponent application using ADPAC, NPSS, and a controller program; and an aviation safety application running about 100 jobs in near real time. The team has successfully demonstrated (1) a Common-Object-Request-Broker-Architecture- (CORBA-) to-Globus resource manager gateway that allows CORBA remote procedure calls to be used to control the submission and execution of programs on workstations and massively parallel computers, (2) a gateway from the CORBA Trader service to the Grid information service, and (3) a preliminary integration of CORBA and Grid security mechanisms. We have applied these technologies to two applications related to NPSS, namely a parameter study and a multicomponent simulation.

INTRODUCTION

Within NASA's High Performance Computing and Communication (HPCC) Program, the NASA Glenn Research Center at Lewis Field is developing an environment for analyzing and designing aircraft engines called the Numerical Propulsion System Simulation (NPSS) (ref. 1). NASA's vision for NPSS is to create a "numerical test cell," enabling full engine simulations overnight on cost-effective computing platforms. To this end, NPSS integrates multiple disciplines such as aerodynamics, structures, and heat transfer, and it supports "numerical zooming" from zero-dimensional to one-, two-, and three-dimensional component engine codes. To facilitate the timely and cost-effective capture of complex physical processes, NPSS uses object-oriented technologies such as C++ objects to encapsulate individual engine components and Object Request Brokers (ORB's) from the Common Object Request Broker Architecture (CORBA) for object communication and deployment across heterogeneous computing platforms.

Recently, the HPCC and Base R&T programs initiated a concept called the Information Power Grid (IPG, ref. 2), a virtual computing environment that integrates computers and other resources at different sites (ref. 3).

IPG implements a range of Grid services such as resource discovery, scheduling, security, instrumentation, and data access, many of which are provided by the Globus toolkit (ref. 4). IPG facilities have the potential to benefit NPSS considerably. For example, NPSS should, in principle, be able to use Grid services to dynamically discover and then coschedule the resources required for a particular engine simulation, rather than relying on manual placement of ORB's as at present. Grid services can also be used to initiate simulation components on massively parallel processors (MPP's) and to address intersite security issues that currently hinder the coupling of components across multiple sites.

These considerations led Glenn personnel and Globus project personnel at Argonne National Laboratory to formulate a collaborative project designed to evaluate whether and how benefits such as those just listed can be achieved in practice. This project involves, first, the development of the basic techniques required to achieve the coexistence of commodity object technologies and Grid technologies, and second, the evaluation of these techniques in the context of NPSS-oriented challenge problems.

The work on basic techniques seeks to understand how commodity technologies (CORBA, DCOM, MS Excel (Microsoft Corp.), etc.) can be used in concert with specialized Grid technologies (for security, MPP scheduling, etc.). In principle, this coordinated use should be straightforward because of the Globus and IPG philosophy of providing low-level Grid mechanisms that can be used to implement a wide variety of application-level programming models. (Globus technologies have previously been used to implement Grid-enabled message-passing libraries, collaborative environments, and parameter study tools, among other things.) The results obtained to date are encouraging: a CORBA-to-Globus resource manager gateway has been successfully demonstrated that allows the use of CORBA remote procedure calls to control the submission and execution of programs on workstations and MPP's; a gateway has been implemented from the CORBA Trader service to the Grid information service; and a preliminary integration of CORBA and Grid security mechanisms has been completed.

The following challenge problems were considered:

(1) *Desktop-controlled parameter study*: Here, an Excel spreadsheet would be used to define and control a computational fluid dynamics (CFD) parameter study via a CORBA interface to a high-throughput broker that would run individual cases on different IPG resources.

(2) *Multicomponent application*: Here, three distinct components—ADPAC, NPSS, and a controller program—would be launched (on workstations or MPP's) and controlled via Globus mechanisms. The components then would communicate among themselves using CORBA.

(3) *Aviation safety*: Here, ~100 near-real-time jobs running NPSS would be submitted and run and data returned in near real time.

In our work to date we have obtained preliminary results for the first two of these problems. This paper presents the following information:

(1) A detailed analysis of the requirements that NPSS applications place on IPG.

(2) A description of the techniques used to meet these requirements via the coordinated use of CORBA and Globus.

(3) A description of the results obtained to date for the first two challenge problems.

(4) The evaluation criteria used to report the results include the time to port, the execution time, the potential scalability of the simulation, and the reliability of the resources.

NPSS AND THE GRID

NPSS is interested in creating an architecture that adopts standards, or application program interfaces (API's) that can assemble engine simulations via a building-block approach. In this way, the NPSS architecture will be able to take advantage of, or "leapfrog" to, the best ideas without redoing the architecture, and it will do this in the shortest time possible. NPSS did this when it adopted the object-oriented paradigm (leveraging its reusability and extensibility features) and when it adopted CORBA for moving objects around a distributed computing simulation. In addition, this approach has been used successfully in building a computer-aided design (CAD) API called CAPRI for a common access to geometry within the NPSS architecture.

The NPSS roadmap was followed to first create a zero-dimensional engine system written in C++ and based on object-oriented design. Designed into this system were the appropriate objects to assemble a multifidelity, multidisciplinary engine analysis capable of accessing engine component codes across different computing platforms. This was and is the power of object-oriented design. As NPSS progressed from concentrating on the zero-dimensional analysis and began to define the required architecture to assemble one-, two-, and three-dimensional codes, the need for batch scheduling software emerged as a focused requirement. Although batch-scheduling software has existed for some time (e.g., PBS, LSF (platform computing), and Condor (University of Wisconsin)), NPSS has never been in a position to dictate one piece of software over another. Indeed, to create a simulation using the most desirable codes, NPSS must create an architecture that does not exclude the use of certain one-, two-, or three-dimensional codes simply because those codes use a piece of software that is incompatible with the architecture. What is required is portability, which is often defined as software that can run everywhere. Not only does NPSS agree with this position, but it further defines this concept to require software to reach everywhere. If a particular piece of software executes on only one platform, NPSS should not force the conversion of that code to some favorable computing platform. Rather, NPSS should provide a means to reach that platform—that is, the power and flexibility of an object-oriented design that can be implemented in C++, CORBA, and now the Grid.

The NPSS project is interested in the Grid specifically as a transparent means to access the different platforms required for executing the various codes that comprise specific NPSS simulations. As we envision it, the task of providing access embraces a wide range of problems, including resource discovery, authentication, authorization, potential privacy of data, executable staging, scheduling, and computation monitoring and control.

The heavy use made of CORBA within NPSS introduces another set of concerns heretofore not encountered in developing and using Grid resources. Although CORBA provides numerous attractive features for developers of complex systems such as NPSS, CORBA implementations are not typically constructed to support the specialized resources encountered in Grid environments (e.g., MPP's and high-speed networks) or to exploit the specialized services provided by Grid systems such as NASA's IPG (e.g., public key authentication). The effective use of Grid concepts within NPSS requires that methods be found that will allow CORBA and Grid services to coexist.

NPSS' INTEREST IN GLOBUS—PORTABILITY, SECURITY, AND REDUCED TURNAROUND TIME

Portability

Globus came along at the same time NPSS started to formally design the one-, two-, and three-dimensional object infrastructure required to assemble the aerospace CFD, structural, thermal, and acoustic codes that use schedulers such as PBS, LSF, and Condor. Within NPSS' definition of portability, Globus provides a leapfrogging technique. Assembling a simulation comprising not only multifidelity, multidiscipline codes but also multiple batch schedulers would not be possible without a tool like Globus. NPSS strives to be scheduler indifferent by adopting appropriate API's to build on. This should result in stability and extensibility within the NPSS architecture.

Security

NPSS requires a security infrastructure that can span multiple administrative domains. Its need to reach everywhere indicates that different NPSS components may need to run on, and communicate between, resources that execute at different sites. Each resource may be governed by site-specific policies and procedures for remote access and use. The Globus Security Infrastructure is designed specifically for this sort of multi-institutional, distributed computing environment. It provides features such as single sign-on access to resources spanning multiple administrative domains, automatic mapping to local accounts and security mechanisms (e.g., Kerberos) within a domain, and delegation of security credentials so that NPSS components running on various resources can act on the user's behalf when authenticating each other and when accessing storage and other resources.

Reduced Turnaround Time

Glenn researchers have been pursuing cost-effective computing technology for some time. Globus' ability to be scheduler indifferent in finding available resources for NPSS simulations helps to reduce or maintain a required overall simulation turnaround time. Without this, certain computing resources known only to the individual schedulers may become overloaded, exhausting NPSS' available resource pool and making the system unable to maintain a desired simulation turnaround time. Globus offers NPSS an extension into the available pool of computing platforms outside a particular scheduler's domain. In the future, there may be a need to cluster Palm Pilots together into a network-addressable secure computing platform. The technology and knowledge to do just that are currently available, and Globus could reduce turnaround time here as well.

Understand that the NPSS architecture has been tasked with supporting simulations that must minimally execute to a solution overnight on cost-effective computing platforms regardless of the complexity of the assembled simulation. As the architecture matures, the simulation execution time must be reduced even further, approaching realistic time, if not real time. Positioning to use tools like Globus is an appropriate strategy for NPSS.

COMMODITY GRID TOOLKITS: INTEGRATING COMMODITY TECHNOLOGIES AND THE GRID

The NPSS project's interest in using Grid and Globus services within a commodity context (CORBA in this case) meshed well with the research goals of the Commodity Grid Toolkits (CoG Kits) project being conducted by the Globus project team. In concept, the notion of a CoG Kit is straightforward: it defines and implements a set of general components that map Grid functionality into a commodity environment or framework (ref. 5), allowing for example, an application to be expressed in terms of familiar CORBA concepts and services, while still exploiting the specialized services (e.g., security and resource discovery) provided by the underlying Grid environment. In practice, defining appropriate components and mappings is far from trivial and, indeed, can raise challenging research issues.

To date, CoG Kit project participants have developed a preliminary Java CoG Kit, which has already proved useful in a variety of settings, and have prototyped elements of a CORBA CoG Kit. It is the latter that we exploit in the work described here. In brief, the work with the CORBA CoG Kit addressed the following issues:

(1) *CORBA Object Creation on Schedulers, via GRAM*: A CORBA gateway to the Globus service was prototyped that supports remote submission, monitoring, and control of computations: the Globus Resource Allocation Manager (GRAM). CORBA clients can connect to this object—via a CORBA Inter Object Reference (IOR), a Naming service, or another way—and then delegate credentials, submit a job, or monitor or cancel a job. Also, clients can receive state change events from this object. The last of these items illustrates how a CoG Kit can translate between Grid and commodity concepts: our gateway takes the GRAM state change events and turns them into events that are passed back to the CORBA client by using a CORBA Event service. We have used this service to initiate and then control, for example, parallel programs on parallel computers.

(2) *Naming Service*: An implementation of a CORBA Naming service was developed on the basis of the Lightweight Directory Access Protocol (LDAP)-based Metacomputing Directory Service (MDS) used in Globus. The CORBA Naming service takes a hierarchichal name and returns an object reference. The Naming service uses the MDS/LDAP distinguished name (DN) as the name, looks for a "corbaIOR" attribute in the object class referred to by the DN, and returns an object reference using the IOR. The Naming service, when combined with the GRAM/CORBA interface, makes it possible to construct the following interesting capability. A site could start up a CORBA-to-GRAM gateway and then add the corbaIOR attribute to the ResourceManager object class in MDS (this is the class that is used by GRAM to describe GRAM resource managers). Now a CORBA client can use our Naming service to go to the augmented ResourceManager object in MDS to retrieve the IOR to be used to submit a job to that resource manager from the CORBA client.

(3) *Trading Service*: A gateway was prototyped that extracts information about resources taken from the Globus MDS and publishes this information into a standard CORBA Trading service. This gateway allows a CORBA client to search the Trading service to find an appropriate resource manager, obtain an object reference to it (using the corbaIOR attribute), and submit a job using the GRAM gateway. An alternative approach would be to implement a Trading service that uses MDS directly, by mapping Trading service queries into MDS queries. This approach, however, poses the challenge of mapping Trading service search constraints to LDAP search filters. We plan to explore this approach in the future.

DESKTOP-CONTROLLED PARAMETER STUDY

The objective of the desktop-controlled parameter study was to develop the CORBA infrastructure support within Globus. The NPSS V1.0 code was used for this demonstration. The NPSS code is a zero-dimensional aerothermodynamic engine model that also has a number of design features for zooming and multidisciplinary coupling.

However, used in its simplest form, NPSS V1.0 provides the performance of a given engine over its flight regime in both steady-state and transient operations. In characterizing an engine's performance, hundreds to thousands of runs of NPSS V1.0 must be executed. To enable the execution of all these jobs, NPSS and the Argonne team experimented with the use of a Globus-based system called the High Throughput Broker (HTB), which supports the mapping of a collection of tasks to Globus-accessible resources that can handle, for example,

- The discovery of these resources, via MDS
- The staging of executables to those resources, via the Global Access to Secondary Storage (GASS) service
- Submission and monitoring of remote computations, via GRAM

Using Globus to deploy 100 to 300 NPSS V1.0 jobs and returning all these results back into one time step presents an attractive use of Globus for engine design studies. For the sample demonstration, 100 to 300 NPSS V1.0 jobs were initiated from an Excel spreadsheet. The individual NPSS jobs communicated between COM (Microsoft Corp.) and CORBA and the Globus HTB service to deploy, manage, and return results to the spreadsheet for later analysis with Excel's graphing and editing features. This is represented pictorially in figure 1.

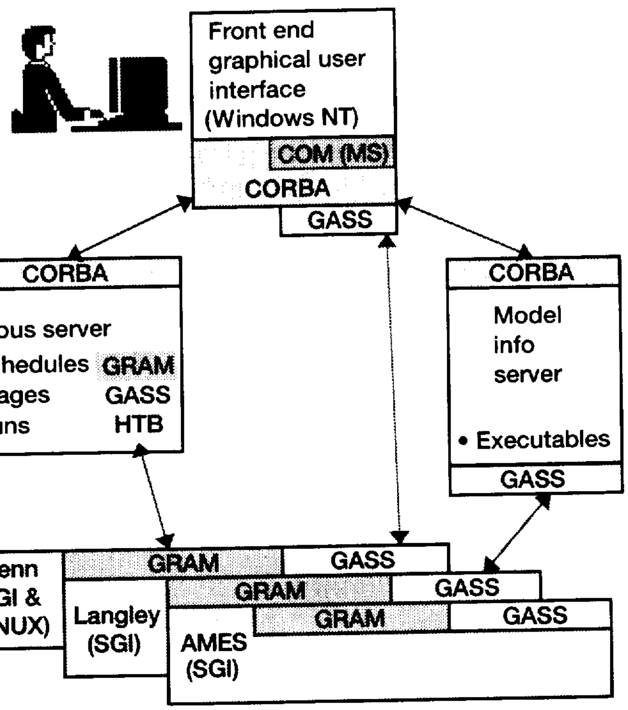

Figure 1.—CORBA and Globus. (CORBA, Common Object Request Broker Architecture; MS, Microsoft; GRAM, Globus Resource Allocation Manager; GASS, Global Access to Secondary Storage; HTB, High Throughput Broker; SGI, Silicon Graphics, Inc.)

Results of the Parameter Study

The creation of the ModelInfoServer and the COM/CORBA bridge took about 2 weeks. This time included both experimentation with various created versions and debugging. Most of the time was taken up with interfacing Excel to the created COM object (via Visual Basic for Applications (VBA, Microsoft Corp.)) and with installing and configuring a working version of Globus on the machines for testing. For much of the development, the HTB ended up being bypassed so that the rest of the simulation could be tested.

The rapid pace of this testing and development was achieved by using Python. Python is an interpreted, interactive, object-oriented high-level programming language with dynamic semantics. It is often compared with Tcl, Perl, Scheme, or Java. Its high-level built-in data structures, combined with dynamic typing and dynamic binding, make it very attractive for rapid application development, as well as for use as a scripting or "glue" language to connect existing components.

Python supports CORBA and provides COM integration. The first version of the ModelInfoServer, which was a CORBA server, was coded in 1 day in C++. For portability (languages like Python and Perl are far more portable than C++ and Java) and ease of modification, it was recoded in Python. This took but a couple of hours. The COM integration was even more astonishing. Via some Python and COM magic, a Python class can be turned into a COM server via the inclusion of a few variables, for things such as the COM classid and a registration function. Most important, this code does not get in the way of the regular functioning of the Python class. Thus, a non-COM test suite could be easily developed, and the same class used to control the simulation could be used in a non-COM/non-Windows environment. Testing and debugging the COM server object or the regular Python class was merely the difference between

```
#Create and attach to a COM Object registered as "NPSSDemo"
o=win32com.client.Dispatch("Python.NPSSDemo")
```

and

```
#Create an instance of NPSSDemo
o=NPSSDemo()
```

The functionality of the COM/CORBA bridge consisted mostly of adapting the CORBA interfaces to COM interfaces. This included mapping function calls and massaging data from one format to another. The COM/CORBA layer included the so-called business logic, that is, the place where the work gets done. Thus, some functions were easy one-to-one mappings:

```
def pauseRun():
        self.session.pause()
def getSession(self):
        if not self.session:
            print "...creating a session"
            status,session=self.fact.create(self.sessionName)

            if status != HTB.SUCCESS:
                raise 'could not create session'

            if session is None:
                raise 'session is None'

            self.session=session

        return self.session
```

whereas others massaged the data, as in the following example. Here the data from the infoServer.listModels command is actually returned as an array of structures that VBA cannot understand, so we put the relevant information into a list and returned it to VBA through COM.

```
def getModelNames(self):
        self.safeConnect()
        shortModelList=self.infoServer.listModels()
        ret=[] #creating a new empty list
        for item in shortModelList:
                ret.append(item.name)
        return ret
```

As stated before, the first incarnation of this was done in C++, and although the wonderful tools provided with Visual C++ (Microsoft Corp.) make it easy to create a COM object, the rapid development and portability of the Python approach won out.

Execution Time

Execution was a little slow. Some of it could be attributed to the actual staging of files, as files would go from the location of the ModelInfoServer to the location of the HTB, then finally to the final destination of the machine. Each transfer, which was composed of 19 files, was around 28 MB. In addition, one special case file was generated per task executed. The size of the per-task case file was usually less than 1 kB. If we were running 100 jobs, we would have 119 files, which would be 28 MB + 100(<1 kB). Since the purpose of this first demonstration was to explore the benefits provided by CORBA coupling, the focus was on functionality rather than on speed.

Scalability and Reliability

Since Globus resources, predominantly the HTB, are responsible for staging and running the jobs, scalability and reliability are essentially guaranteed. The issues will probably resolve as we determine how long it takes the HTB to pump out jobs and how many sustained jobs it can maintain. However, these will all be handled at the HTB level.

Although problems arose in getting a usable version of Globus 1.1 up and running on our machines, they have all been resolved.

MULTICOMPONENT APPLICATION

The objective of this demonstration was to deploy a mixed-fidelity engine simulation over Globus. Two codes, ADPAC and NPSS V1.0, were chosen to simulate the complete low-pressure system of the Energy Efficient Engine. ADPAC, an MPI-based code, modeled the low-pressure subsystem in three dimensions, whereas NPSS V1.0 modeled the engine core in zero dimensions. The two codes communicated with each other through CORBA. This simulation was developed to measure the sophistication of Globus' ability to deploy a mixed-fidelity, MPI CORBA simulation. The engineering purpose of the simulation was to determine the revolutions per minute at which the power required by the fan was balanced by the power available from the low-pressure turbine. The entire engine was simulated at different fidelity levels depending on the required accuracy. Figure 2 shows the multifidelity engine model simulation. Three-dimensional CFD analysis of the low-pressure subsystem was performed by two instances of ADPAC represented by the dark shaded area in the figure, whereas NPSS V1.0 was used for a cycle (zero-dimensional) analysis of the core represented by the light shaded area in the figure. The ADPAC code solves a

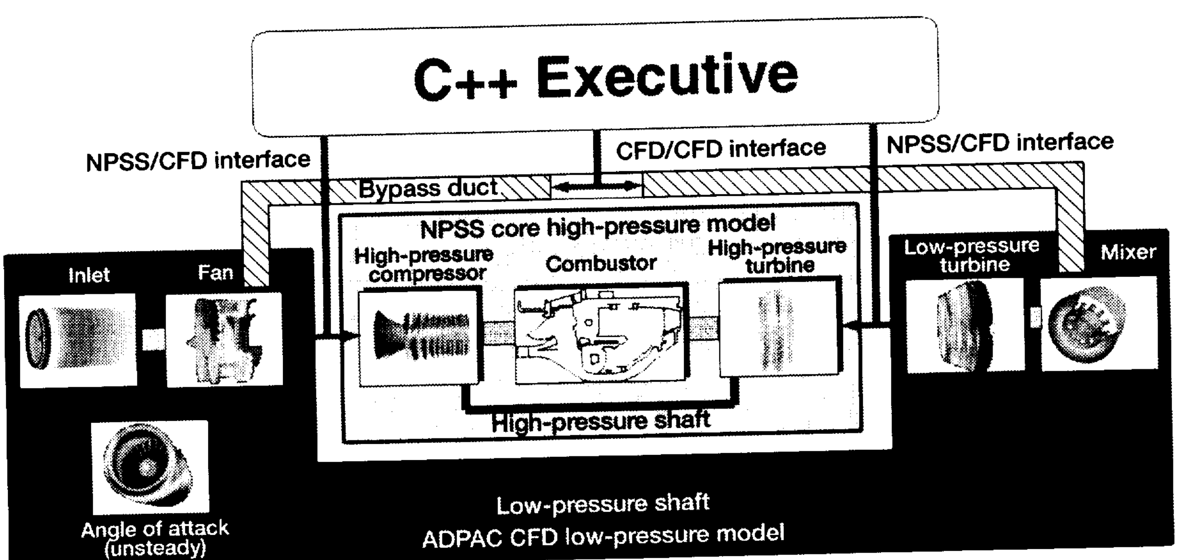

Figure 2.—Current computational multifidelity engine model. Computational-fluid-dynamics- (CFD-) based shaft power balance: 20 processors, 5-rpm adjustments, 14 hr.

three-dimensional problem of the Reynolds-averaged Navier-Stokes equations using a time-marching, finite volume algorithm. It employs a flexible multiple-block structured grid discretization scheme with user-configurable boundary conditions. This package provides a flexible aerodynamic simulation environment for complex compressible flows, and numerous features to simulate multistage turbomachinery flows. ADPAC uses coarse-grained domain decomposition for parallel processing with interprocessor message passing. This code has been validated for a wide variety of propulsion flow applications including multistage compressor and turbine turbomachinery predictions for inlets, nozzles, and propellers.

The demonstration at Supercomputing'99 (SC'99) was run across various NASA IPG hosts via the Globus utilities "globus-gass-server" and "globusrun." CORBA facilitated communication between the various analysis codes, whereas MPI handled communication within the parallel CFD codes. A Java executive implemented a simple iterative solver to determine the revolutions per minute and provided a graphical user interface to view the simulation's progress.

Results of the Multicomponent Application

A full evaluation of this simulation was not completed in time to report it within this paper. Suffice it to say that the simulation passed a believability toll-gate in that Globus did not interfere with this simulation even though minimal Globus features were exercised to this point. Table I characterizes the results obtained for this application.

TABLE I.—RESULTS OF THE MULTICOMPONENT APPLICATION

Time to port	Execution time	Scalability	Reliability
Hard to measure—must include learning Globus and acquiring security ticketing; after this, time to port is minimal	Similar to any batch queuing system—sluggish at startup, then no difference	For particular jobs, very scalable; for those that require intense messaging, not good	No different from any current batch queuing system— LSF, PBS, etc.

SUMMARY OF RESULTS

This project was designed to evaluate the feasibility of combining "Grid" and "commodity" technologies, with a view to leveraging the numerous advantages of commodity technologies in a high-performance Grid environment. Results obtained to date are encouraging. The team has successfully demonstrated (1) a CORBA-to-Globus resource manager gateway that allows CORBA remote procedure calls to be used to control the submission and execution of programs on workstations and massively parallel computers, (2) a gateway from the CORBA Trader service to the Grid information service, and (3) a preliminary integration of CORBA and Grid security mechanisms. We have applied these technologies to two applications related to NPSS: namely, a parameter study and a multicomponent simulation.

In future work, we plan to build on these foundations to enable complex simulations to be solved in quasi-real time. As an example, NPSS will support the Aviation Safety Program concept of modeling the National Airspace System. In this simulation, the NPSS V1.0 code will be deployed onto a Globus computing platform that can process 2000 to 3000 flights per day. NPSS V1.0 will accept flight data for airline departures, routes, and landings from a major U.S. airport currently sized to handle 3000 flights per day. Flight data will be transmitted to Glenn from the NASA Ames Research Center over a web-based CORBA connection where NPSS V1.0 will process flight data and return an appropriate number of engine performance and risk assessment parameters to Ames. NPSS expects to handle 5000 to 6000 engine models per day in realistic to real time.

REFERENCES

1. Evans, A.L., et al.: An Integrated Computing and Interdisciplinary Systems Approach to Aeropropulsion Simulation. ASME Paper 97–GT–303, 1997.
2. Johnston, W.E.; Gannon, D.; Nitzberg, B.: Grids as Production Computing Environments: The Engineering Aspects of NASA's Information Power Grid. Proceedings of the Eighth High Performance Distributed Computing Symposium, IEEE Computer Society Press, Los Alamitos, CA, 1999.

3. Foster, I.; and Kesselman, C.: The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann Publishers, San Francisco, CA, 1999.
4. Foster, I.; and Kesselman, C.: The Globus Project: A Status Report. Proceedings of the Heterogeneous Computing Workshop, IEEE Computer Society Press, 1998, pp. 4–18.
5. von Laszewski, G., et al.: CoG Kits: A Bridge Between Commodity Distributed Computing and High-Performance Grids. ACM Java Grande 2000 Conference, San Francisco, CA, June 3–5, 2000.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*

2. REPORT DATE
July 2000

3. REPORT TYPE AND DATES COVERED
Technical Memorandum

4. TITLE AND SUBTITLE
NPSS on NASA's Information Power Grid: Using CORBA and Globus to Coordinate Multidisciplinary Aeroscience Applications

5. FUNDING NUMBERS
WU–509–10–24–00

6. AUTHOR(S)
Isaac Lopez, Gregory J. Follen, Richard Gutierrez, Ian Foster, Brian Ginsburg, Olle Larsson, Stuart Martin, Steven Tuecke, and David Woodford

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
NASA Glenn Research Center
Cleveland, Ohio 44135–3191
and
U.S. Army Research Laboratory
Cleveland, Ohio 44135–3191

8. PERFORMING ORGANIZATION REPORT NUMBER
E–12212

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Washington, DC 20546–0001
and
U.S. Army Research Laboratory
Adelphi, Maryland 20783–1145

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
NASA TM—2000-209956
ARL–TR–2262

11. SUPPLEMENTARY NOTES
Prepared for the Computational Aerosciences Workshop sponsored by the High Performance Computing and Communications Program, Moffett Field, California, February 15–17, 2000. Isaac Lopez, U.S. Army Research Laboratory, Glenn Research Center; Gregory J. Follen, NASA Glenn Research Center; Richard Gutierrez, Integral Systems, Cleveland, Ohio 44135; Ian Foster, Brian Ginsburg, Olle Larsson, Stuart Martin, Steven Tuecke, and David Woodford, Argonne National Laboratory, 9700 S. Cass Avenue, Argonne, Illinois 60439. Responsible person, Isaac Lopez, organization code 0300, (216) 433–5893.

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified - Unlimited
Subject Categories: 01 and 61 Distribution: Nonstandard

This publication is available from the NASA Center for AeroSpace Information, (301) 621–0390.

12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 words)***

This paper describes a project to evaluate the feasibility of combining Grid and Numerical Propulsion System Simulation (NPSS) technologies, with a view to leveraging the numerous advantages of commodity technologies in a high-performance Grid environment. A team from the NASA Glenn Research Center and Argonne National Laboratory has been studying three problems: a desktop-controlled parameter study using Excel (Microsoft Corporation); a multicomponent application using ADPAC, NPSS, and a controller program; and an aviation safety application running about 100 jobs in near real time. The team has successfully demonstrated (1) a Common-Object-Request-Broker-Architecture-(CORBA-) to-Globus resource manager gateway that allows CORBA remote procedure calls to be used to control the submission and execution of programs on workstations and massively parallel computers, (2) a gateway from the CORBA Trader service to the Grid information service, and (3) a preliminary integration of CORBA and Grid security mechanisms. We have applied these technologies to two applications related to NPSS, namely a parameter study and a multi-component simulation.

14. SUBJECT TERMS
Distributed computing; CORBA; Globus

15. NUMBER OF PAGES
15

16. PRICE CODE
A03

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	